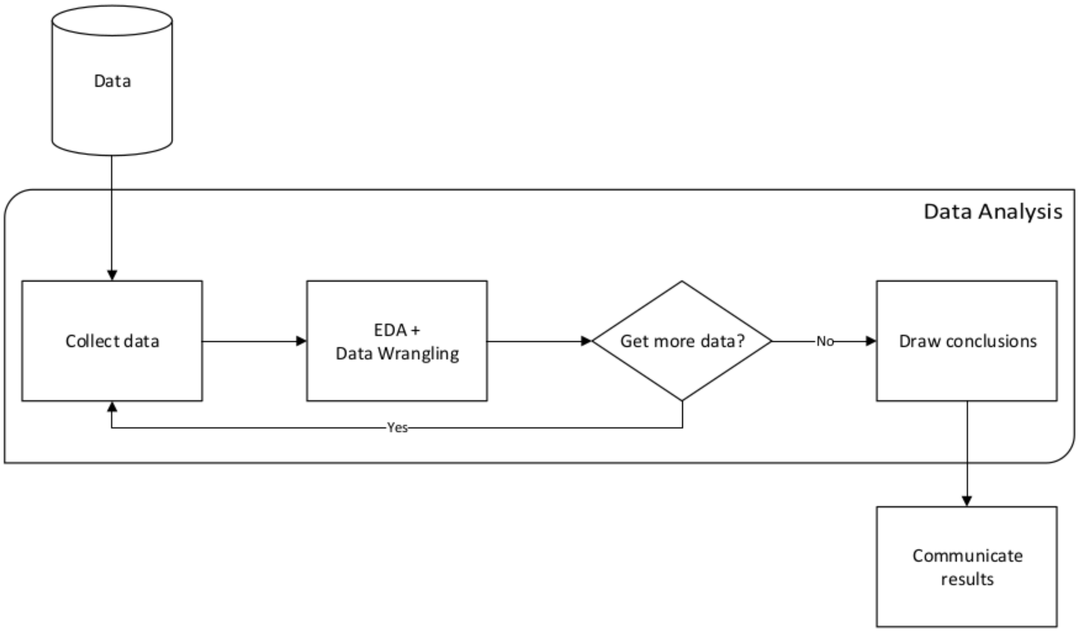
introduction_to_data_analysis

November 30, 2024

1 Introduction to Data Analysis

1.1 The fundamentals of data analysis



1.2 Steps in Data Analysis

- `Data collection`
- `Data wrangling`
- `Exploratory data analysis`
- `Drawing conclusions`

1.2.1 Data collection

Data collection is the natural first step for any data analysis—we can't analyze data we don't have. In reality, our analysis can begin even before we have the data. When we decide what we want to investigate or analyze, we have to think about what kind of data we can collect that will be useful for our analysis. While data can come from anywhere: • Web scraping to extract data from a website's HTML (often with Python packages such as selenium, requests, scrapy, and beautifulsoup) • Application programming interfaces (APIs) for web services from which we can collect data with

HTTP requests (perhaps using cURL or the requests Python package) • Databases (data can be extracted with SQL or another database-querying language) • Internet resources that provide data for download, such as government websites or Yahoo! Finance • Log files

1.2.2 Data wrangling

Data wrangling is the process of preparing the data and getting it into a format that can be used for analysis. The unfortunate reality of data is that it is often dirty, meaning that it requires cleaning (preparation) before it can be used. The following are some issues we may encounter with our data: • Human errors: Data is recorded (or even collected) incorrectly, such as putting 100 instead of 1000, or typos. In addition, there may be multiple versions of the same entry recorded, such as New York City, NYC, and nyc. • Computer error: Perhaps we weren't recording entries for a while (missing data). • Unexpected values: Maybe whoever was recording the data decided to use a question mark for a missing value in a numeric column, so now all the entries in the column will be treated as text instead of numeric values. • Incomplete information: Think of a survey with optional questions; not everyone will answer them, so we will have missing data, but not due to computer or human error. • Resolution: The data may have been collected per second, while we need hourly data for our analysis. • Relevance of the fields: Often, data is collected or generated as a product of some process rather than explicitly for our analysis. In order to get it to a usable state, we will have to clean it up. • Format of the data: Data may be recorded in a format that isn't conducive to analysis, which will require us to reshape it. • Misconfigurations in the data-recording process: Data coming from sources such as misconfigured trackers and/or webhooks may be missing fields or passed in the wrong order.

1.2.3 Exploratory data analysis

During EDA, we use visualizations and summary statistics to get a better understanding of the data. Since the human brain excels at picking out visual patterns, data visualization is essential to any analysis. In fact, some characteristics of the data can only be observed in a plot. Depending on our data, we may create plots to see how a variable of interest has evolved over time, compare how many observations belong to each category, find outliers, look at distributions of continuous and discrete variables, and much more.

- `Data needs to be prepped before EDA.`
- `Visualizations that are created during EDA may indicate the need for additional data cleaning`
- `Data wrangling uses summary statistics to look for potential data issues, while EDA uses them`

1.2.4 Drawing conclusions

After we have collected the data for our analysis, cleaned it up, and performed some thorough EDA, it is time to draw conclusions. This is where we summarize our findings from EDA and decide the next steps:

- `Did we notice any patterns or relationships when visualizing the data?`
- `Does it look like we can make accurate predictions from our data? Does it make sense to move`
- `Should we handle missing data points? How?`
- `How is the data distributed?`
- `Does the data help us answer the questions we have or give insight into the problem we are in`
- `Do we need to collect new or additional data?`

1.3 Setup

[1]:
```
from visual_aids import stats_viz
```

1.4 Statistical Foundations

As this is not a statistics book, we will discuss the concepts we will need to work through the book, in addition to some avenues for further exploration. By no means is this exhaustive.

1.4.1 Sampling

our sample must be a random sample that is representative of the population. This means that the data must be sampled without bias (for example, if we are asking people whether they like a certain sports team, we can't only ask fans of the team) and that we should have (ideally) members of all distinct groups from the population in our sample (in the sports team example, we can't just ask men).

1.4.2 Types of Sampling

- **simple random sampling**: we use a random number generator to pick rows at random.
- **stratified random sampling**: When we have distinct groups in the data, we want our sample to be a stratified random sample, which will preserve the proportion of the groups in the data.
- **bootstrapping**: sampling with replacement (more info: YouTube video and Wikipedia article)

1.4.3 Descriptive Statistics

We use descriptive statistics to describe and/or summarize the data. The data we work with is usually a **sample** taken from the **population**. The statistics we discuss are referred to as **sample statistics** because they are calculated on the sample and can be used as estimators for the population parameters.

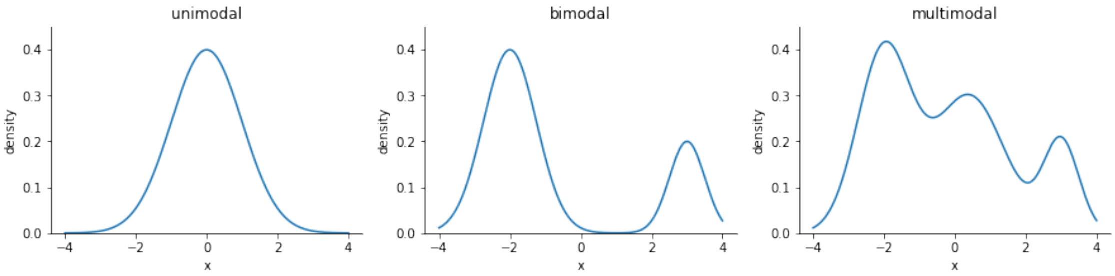
Measures of Center Three common ways to describe the central tendency of a distribution are mean, median, and mode. ##### Mean Perhaps the most common statistic for summarizing data is the average, or mean. The population mean is denoted by (the Greek letter mu), and the sample mean is written as (pronounced X-bar). The sample mean is calculated by summing all the values and dividing bby the count of values. One important thing to note about the mean is that it is very sensitive to outliers. The sample mean is an estimator for the population mean (μ) and is defined as:

$$\bar{x} = \frac{\sum_1^n x_i}{n}$$

Median The median represents the 50th percentile of our data; this means that 50% of the values are greater than the median and 50% are less than the median. Unlike the mean, the median is robust to outliers. It is calculated by taking the middle value from an ordered list of values.

Mode The mode is the most common value in the data. Understanding the concept of the mode comes in handy when describing continuous distributions; however, most of the time when we're describing our continuous data, we will use either the mean or the median as our measure of central tendency. When working with categorical data, on the other hand, we will typically use the mode. For continuous data, the shape of the distribution and as shown in the following plots, a unimodal distribution has only one mode (at 0), a bimodal distribution has two (at -2 and 3), and a multimodal distribution has many (at -2, 0.4, and 3)::

```
[2]: ax = stats_viz.different_modal_plots()
```



Measures of Spread Measures of spread tell us how the data is dispersed; this will indicate how thin (low dispersion) or wide (very spread out) our distribution is.

Range The range is the distance between the smallest value (minimum) and the largest value (maximum). It gives us upper and lower bounds on what we have in the data; however, if we have any outliers in our data, the range will be rendered useless. Another problem with the range is that it doesn't tell us how the data is dispersed around its center; it really only tells us how dispersed the entire dataset is. This brings us to the variance.

$$range = max(X) - min(X)$$

Variance The variance describes how far apart observations are spread out from their average value (the mean). The population variance is denoted as 2 (pronounced sigma-squared), and the sample variance is written as s2. It is calculated as the average squared distance from the mean. When calculating the sample variance, we divide by *n - 1* instead of *n* to account for using the sample mean ($\bar{x}$):

$$s^2 = \frac{\sum_1^n (x_i - \bar{x})^2}{n-1}$$

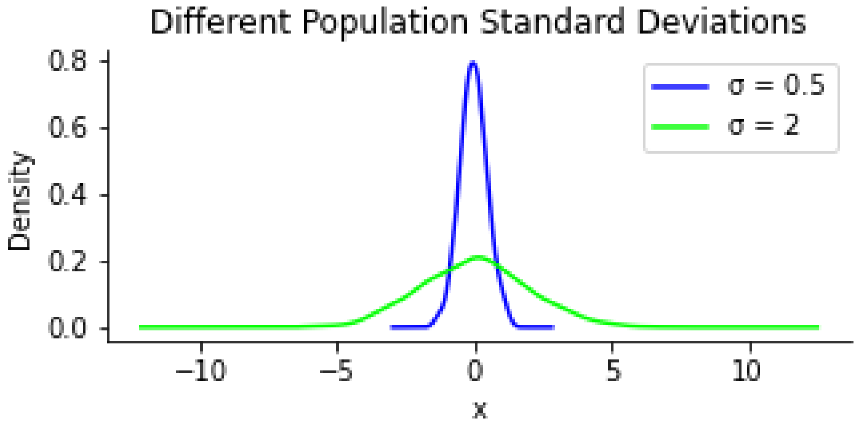
This is referred to as Bessel's correction and is applied to get an unbiased estimator of the population variance.

Note that this will be in units-squared of whatever was being measured.

Standard Deviation The standard deviation is the square root of the variance, giving us a measure in the same units as our data. We can use the standard deviation to see how far from the mean data points are on average. A small standard deviation means that values are close to the mean, while a large standard deviation means that values are dispersed more widely. This is tied to how we would imagine the distribution curve: the smaller the standard deviation, the thinner the peak of the curve (0.5); the larger the standard deviation, the wider the peak of the curve (2). The sample standard deviation is calculated as follows:

$$s = \sqrt{\frac{\sum_1^n (x_i - \bar{x})^2}{n-1}} = \sqrt{s^2}$$

```
[3]: ax = stats_viz.effect_of_std_dev()
```



Note that σ^2 is the population variance and σ is the population standard deviation.

Coefficient of Variation When we moved from variance to standard deviation, we were looking to get to units that made sense; however, if we then want to compare the level of dispersion of one dataset to another, we would need to have the same units once again. One way around this is to calculate the coefficient of variation (CV), which is unitless. The coefficient of variation (CV) gives us a unitless ratio of the standard deviation to the mean. Since, it has no units we can compare dispersion across datasets. The CV is the ratio of the standard deviation to the mean:

$$CV = \frac{s}{\bar{x}}$$

Interquartile Range Median is the 50th percentile or the 2nd quartile (Q2). Percentiles and quartiles are both quantiles—values that divide data into equal groups each containing the same percentage of the total data. Percentiles divide the data into 100 parts, while quartiles do so into four (25%, 50%, 75%, and 100%). Since quantiles neatly divide up our data, and we know how much of the data goes in each section, they are a perfect candidate for helping us quantify the spread of our data. One common measure for this is the interquartile range (IQR), which is the distance between the 3rd and 1st quartiles. The IQR gives us the spread of data around the median and quantifies how much dispersion we have in the middle 50% of our distribution. It can also be useful when checking the data for outliers. The interquartile range (IQR) gives us the spread of data around the median and quantifies how much dispersion we have in the middle 50% of our distribution:

$$IQR = Q_3 - Q_1$$

Quartile Coefficient of Dispersion The quartile coefficient of dispersion also is a unitless statistic for comparing datasets. However, it uses the median as the measure of center. It is calculated by dividing the semi-quartile range (half the IQR) by the midhinge (midpoint between the first and third quartiles):

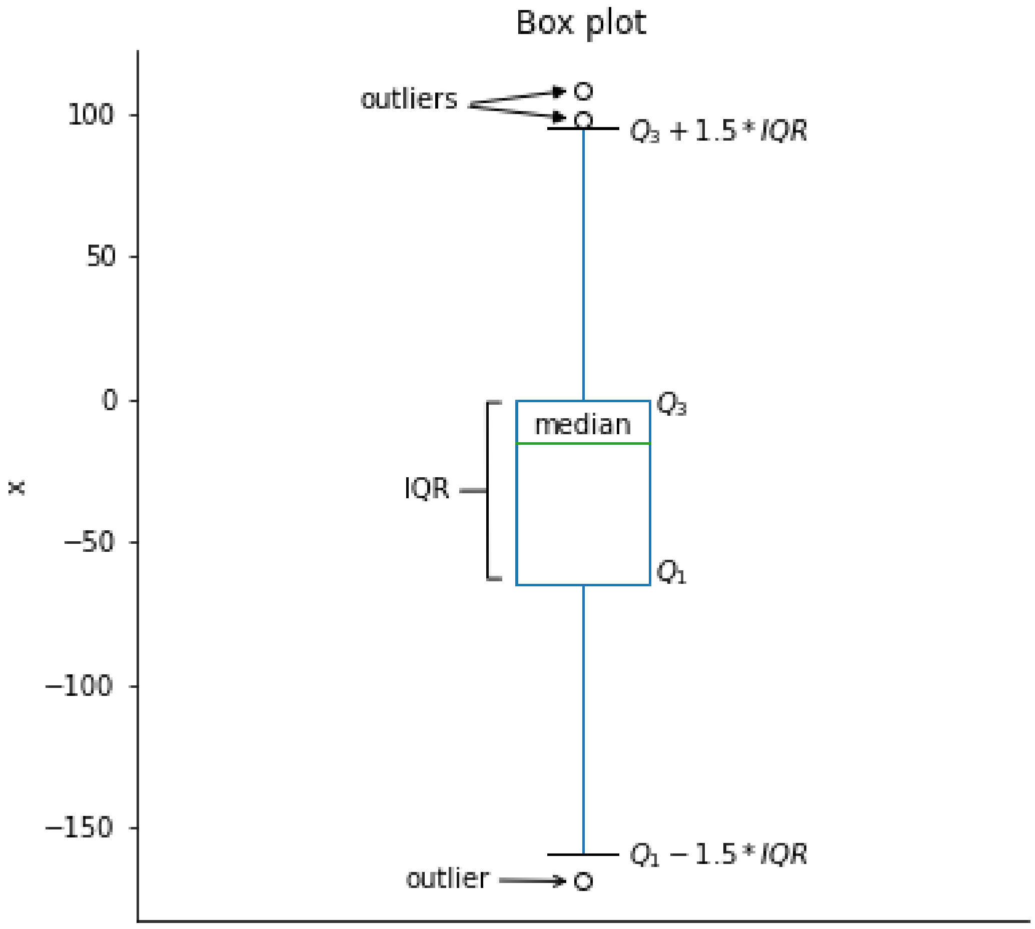
$$QCD = \frac{\frac{Q_3 - Q_1}{2}}{\frac{Q_1 + Q_3}{2}} = \frac{Q_3 - Q_1}{Q_3 + Q_1}$$

Summarizing data The **5-number summary** provides 5 descriptive statistics that summarize our data:

	Quartile	Statistic	Percentile
1.	Q_0	minimum	0^{th}
2.	Q_1	N/A	25^{th}
3.	Q_2	median	50^{th}
4.	Q_3	N/A	75^{th}
5.	Q_4	maximum	100^{th}

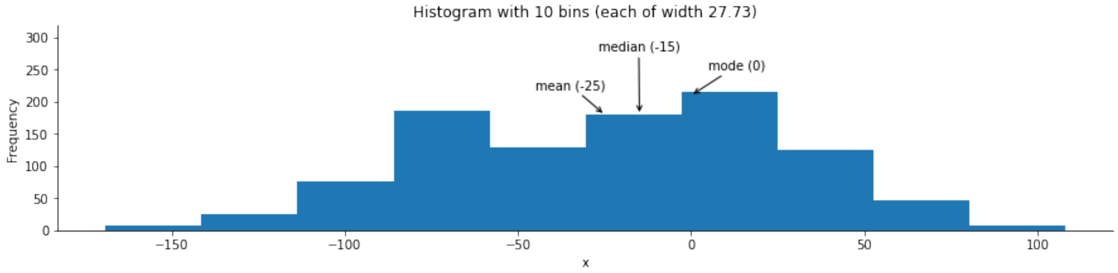
This summary can be visualized using a **box plot** (also called box-and-whisker plot). The box has an upper bound of Q_3 and a lower bound of Q_1. The median will be a line somewhere in this box. The whiskers extend from the box towards the minimum/maximum. For our purposes, they will extend to $Q_3 + 1.5 \times IQR$ and $Q_1 - 1.5 \times IQR$ and anything beyond will be represented as individual points for outliers:

```
[4]: ax = stats_viz.example_boxplot()
```



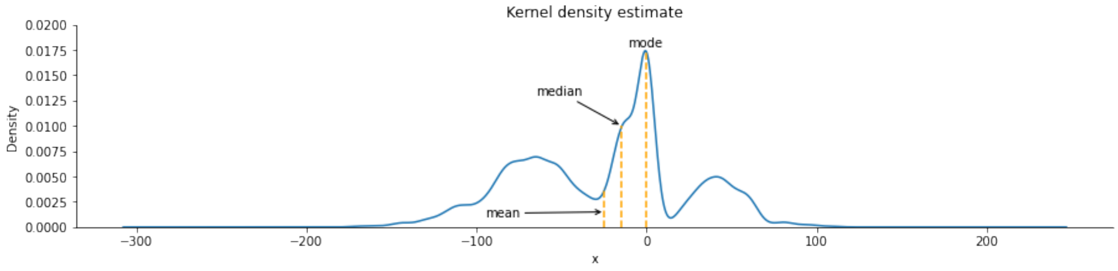
The box plot doesn't show us how the data is distributed within the quartiles. To get a better sense of the distribution, we can use a **histogram**, which will show us the amount of observations that fall into equal-width bins. We can vary the number of bins to use, but be aware that this can change our impression of what the distribution appears to be:

[5]:
```
ax = stats_viz.example_histogram()
```



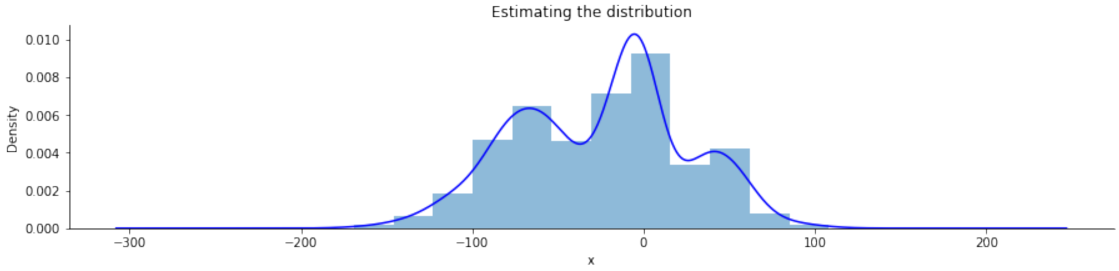
We can also visualize the distribution using a **kernel density estimate (KDE)**. This will estimate the **probability density function (PDF)**. This function shows how probability is distributed over the values. Higher values of the PDF mean higher likelihoods:

```
[6]: ax = stats_viz.example_kde()
```



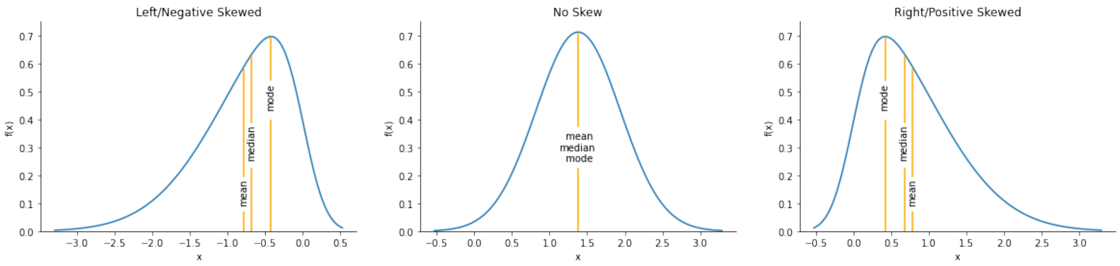
Note that both the KDE and histogram estimate the distribution:

```
[7]: ax = stats_viz.hist_and_kde()
```



Skewed distributions have more observations on one side. The mean will be less than the median with negative skew, while the opposite is true of positive skew:

```
[8]: ax = stats_viz.skew_examples()
```



We can use the **cumulative distribution function (CDF)** to find probabilities of getting values within a certain range. The CDF is the integral of the PDF:

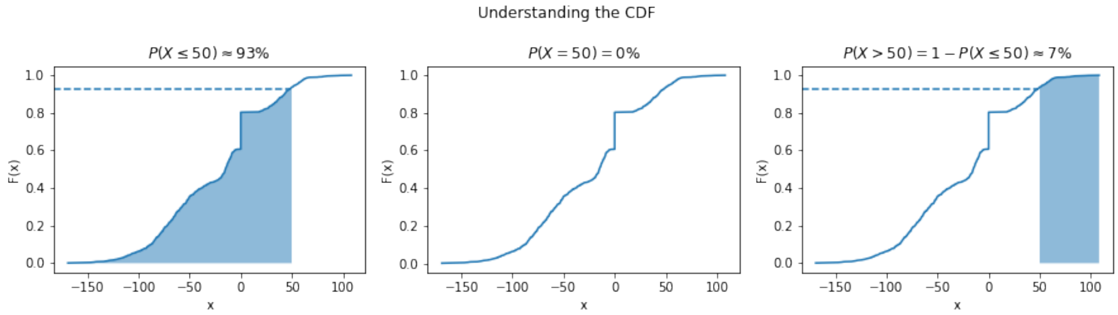
$$CDF = F(x) = \int_{-\infty}^{x} f(t)dt$$

Note that $f(t)$ is the PDF and $\int_{-\infty}^{\infty} f(t)dt = 1$.

The probability of the random variable X being less than or equal to the specific value of x is denoted as $P(X \leq x)$. Note that for a continuous random variable the probability of it being exactly x is zero.

Let's look at the estimate of the CDF from the sample data we used for the box plot, called the **empirical cumulative distribution function (ECDF)**:

```
[9]: ax = stats_viz.cdf_example()
```



We can find any range we want if we use some algebra as in the rightmost subplot above.

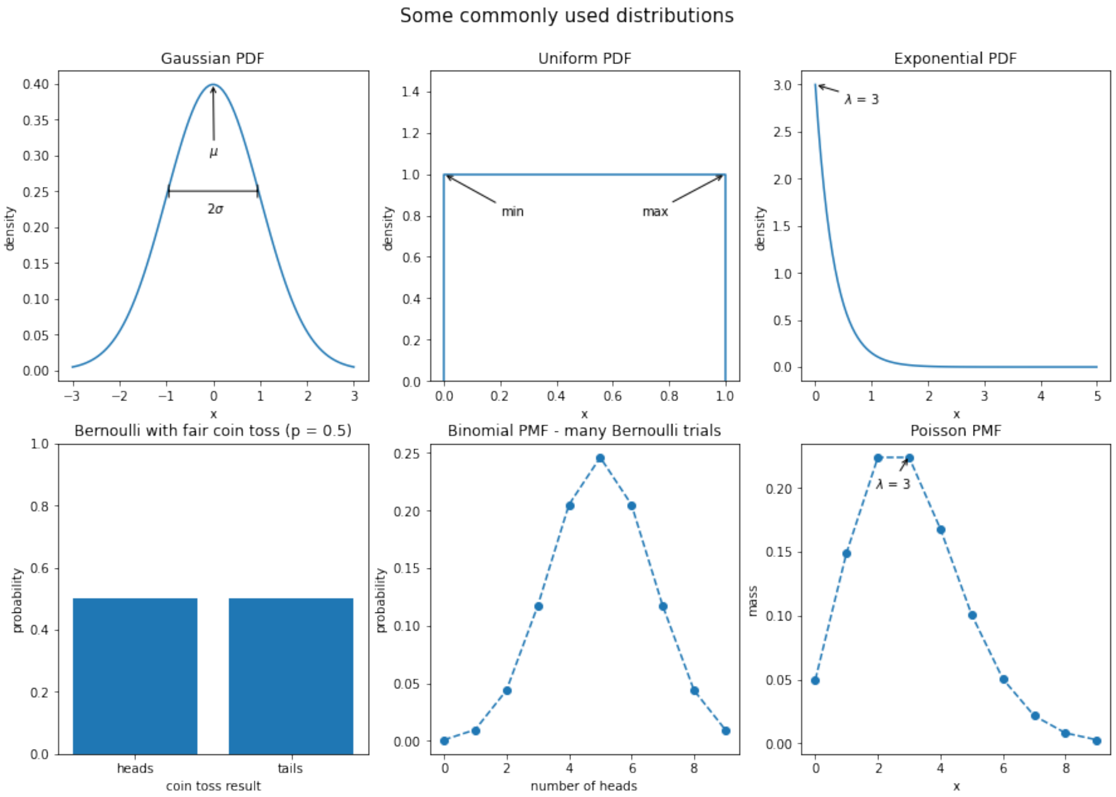
Common Distributions

- **Gaussian (normal) distribution**: looks like a bell curve and is parameterized by its mean ( ) and standard deviation ( ). Many things in nature happen to follow the normal distribution, like heights. Note that testing if a distribution is normal is not trivial. Written as $N(\mu, \sigma)$.
- **Poisson distribution**: discrete distribution that is often used to model arrivals. Parameterized by its mean, lambda ( ). Written as $Pois(\lambda)$.
- **Exponential distribution**: can be used to model the time between arrivals. Parameterized by its mean, lambda ( ). Written as $Exp(\lambda)$.
- **Uniform distribution**: places equal likelihood on each value within its bounds (a and b). We often use this for random number generation. Written as $U(a, b)$.
- **Bernoulli distribution**: When we pick a random number to simulate a single success/failure outcome, it is called a Bernoulli trial. This is parameterized by the probability of success (p). Written as $Bernoulli(p)$.
- **Binomial distribution**: When we run the same experiment n times, the total number of successes is then a binomial random variable. Written as $B(n, p)$.

We can visualize both discrete and continuous distributions; however, discrete distributions give us a **probability mass function** (**PMF**) instead of a PDF:

[10]:

```
ax = stats_viz.common_dists()
```



Scaling data In order to compare variables from different distributions, we would have to scale the data, which we could do with the range by using **min-max scaling**. We take each data point, subtract the minimum of the dataset, then divide by the range. This normalizes our data (scales it to the range [0, 1])::

$$x_{scaled} = \frac{x - min(X)}{range(X)}$$

Z-score: We would subtract the mean from each observation and then divide by the standard deviation to standardize the data. This gives us what is known as a Z-score:. Another way is to use a **Z-score** to standardize the data:

$$z_i = \frac{x_i - \bar{x}}{s}$$

Quantifying relationships between variables The **covariance** is a statistic for quantifying the relationship between variables by showing how one variable changes with respect to another (also referred to as their joint variance):

$$cov(X, Y) = E[(X - E[X])(Y - E[Y])]$$

E[X] is the expectation of the random variable X (its long-run average).

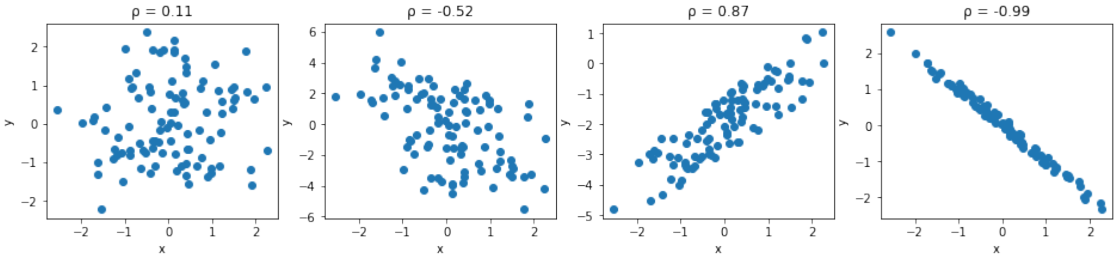
The sign of the covariance gives us the direction of the relationship, but we need the magnitude as well. For that, we calculate the **Pearson correlation coefficient** (ρ):

$$\rho_{X,Y} = \frac{cov(X, Y)}{s_X s_Y}$$

This normalizes the covariance and results in a statistic bounded between -1 and 1, making it easy to describe both the direction of the correlation (sign) and the strength of it (magnitude). Correlations of 1 are said to be perfect positive (linear) correlations, while those of -1 are perfect negative correlations. Values near 0 aren't correlated. If correlation coefficients are near 1 in absolute value, then the variables are said to be strongly correlated; those closer to 0.5 are said to be weakly correlated.

Examples: Let's look at some examples using scatter plots. In the leftmost subplot of Figure 1.12 (= 0.11), we see that there is no correlation between the variables: they appear to be random noise with no pattern. The next plot with = -0.52 has a weak negative correlation: we can see that the variables appear to move together with the x variable increasing, while the y variable decreases, but there is still a bit of randomness. In the third plot from the left (= 0.87), there is a strong positive correlation: x and y are increasing together. The rightmost plot with = -0.99 has a near-perfect negative correlation: as x increases, y decreases. We can also see how the points form a line:

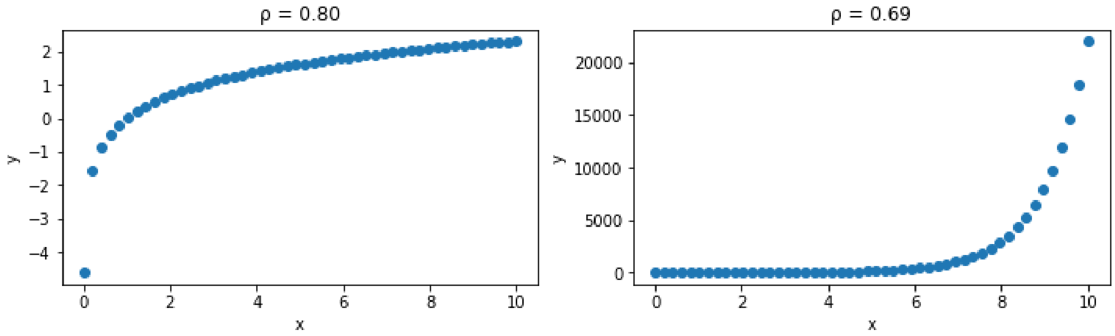
[11]:
```
ax = stats_viz.correlation_coefficient_examples()
```



From left to right: no correlation, weak negative correlation, strong positive correlation, and nearly perfect negative correlation.

Often, it is more informative to use scatter plots to check for relationships between variables. This is because the correlation may be strong, but the relationship may not be linear:

[12]:
```
ax = stats_viz.non_linear_relationships()
```

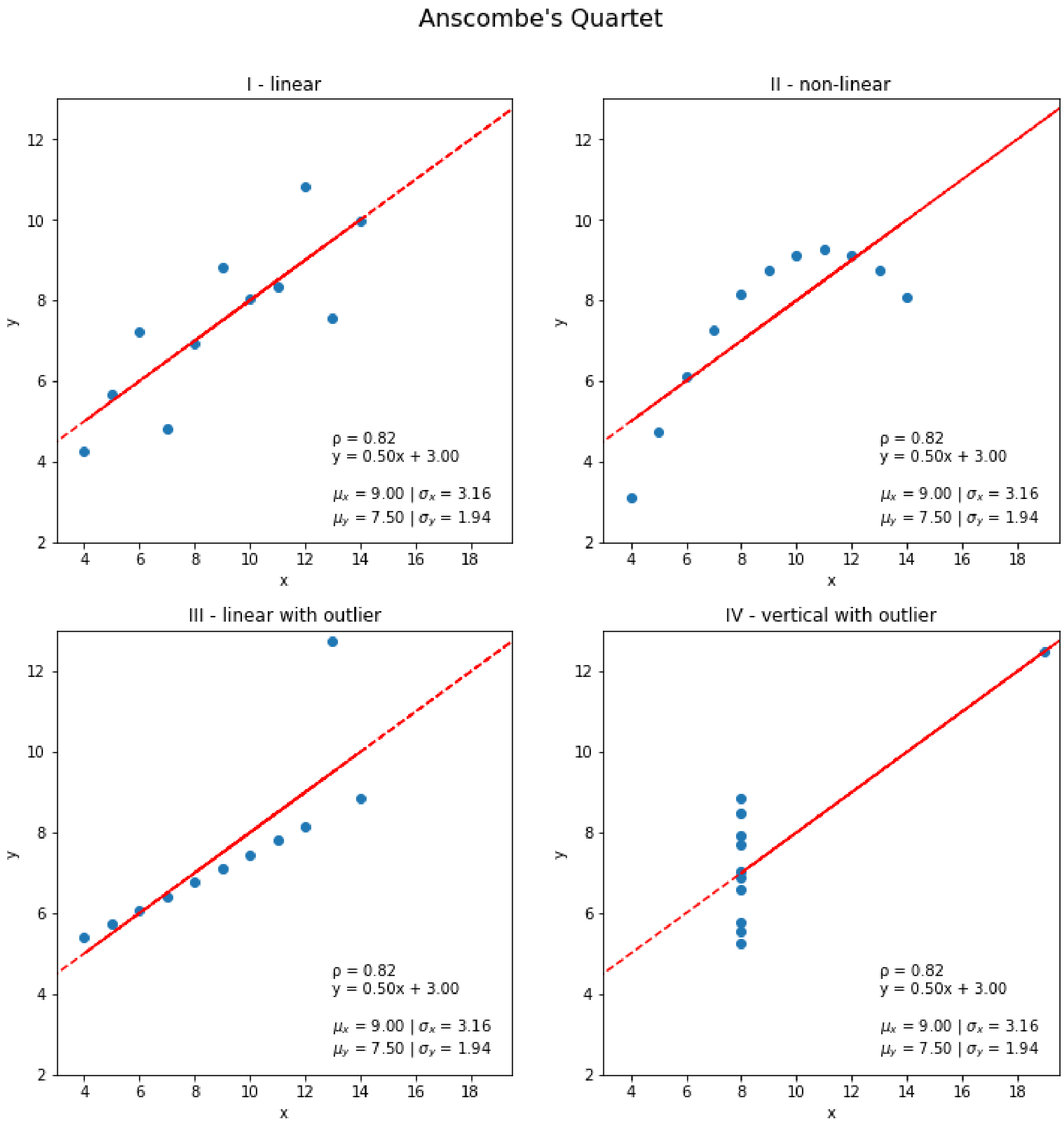


Remember, **correlation does not imply causation**. While we may find a correlation between X and Y, it does not mean that X causes Y or Y causes X. It is possible there is some Z that causes both or that X causes some intermediary event that causes Y — it could even be a coincidence. Be sure to check out Tyler Vigen's Spurious Correlations blog for some interesting correlations.

Pitfalls of summary statistics Not only can our correlation coefficients be misleading, but so can summary statistics. Anscombe's quartet is a collection of four different datasets that have identical summary statistics and correlation coefficients, however, when plotted, it is obvious they are not similar:

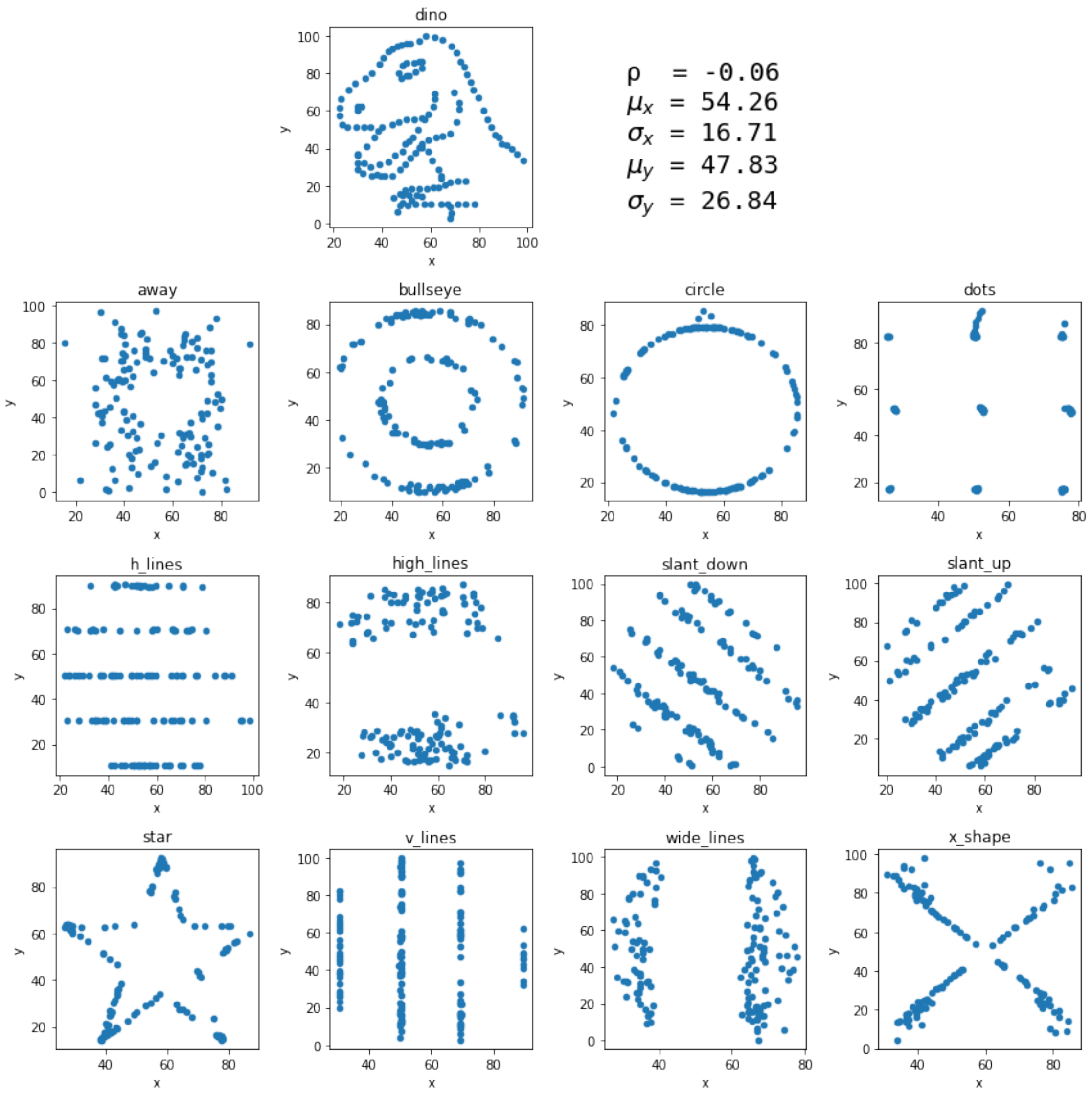
[13]:

```
ax = stats_viz.anscombes_quartet()
```



Another example of this is the Datasaurus Dozen:

[14]:
```
ax = stats_viz.datasaurus_dozen()
```

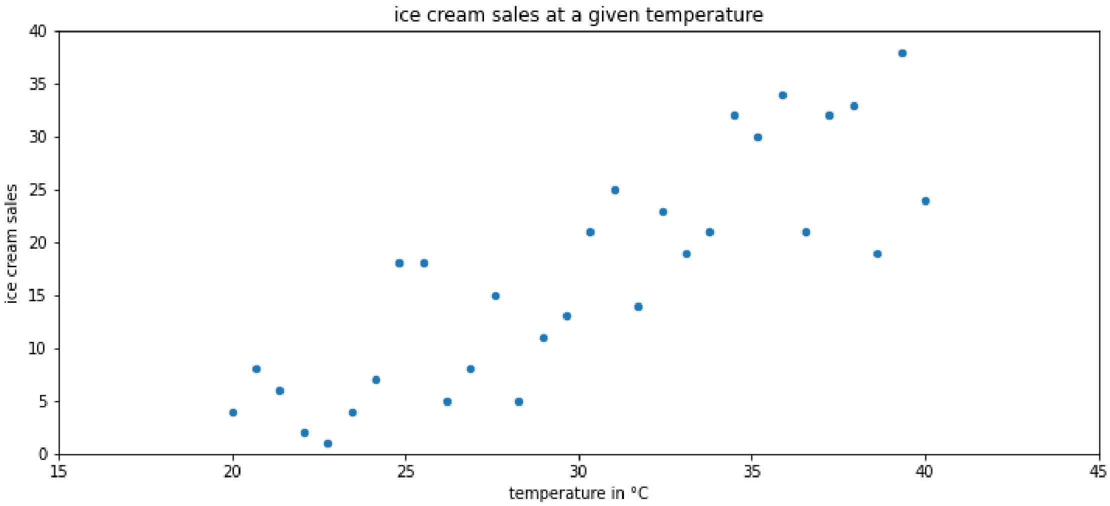


1.4.4 Prediction and forecasting

Say our favorite ice cream shop has asked us to help predict how many ice creams they can expect to sell on a given day. They are convinced that the temperature outside has strong influence on their sales, so they collected data on the number of ice creams sold at a given temperature. We agree to help them, and the first thing we do is make a scatter plot of the data they gave us:

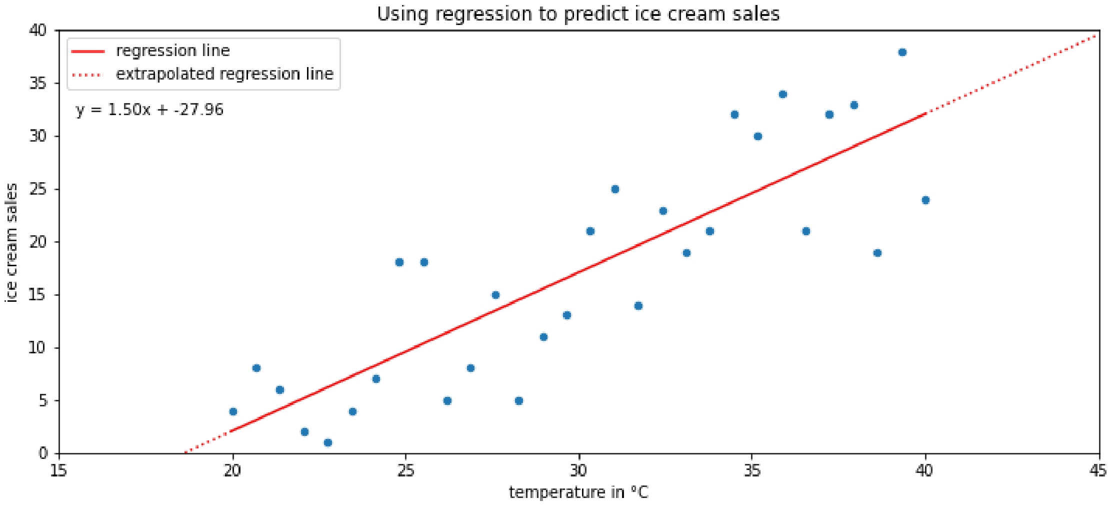
[15]:

```
ax = stats_viz.example_scatter_plot()
```



We can observe an upward trend in the scatter plot: more ice creams are sold at higher temperatures. In order to help out the ice cream shop, though, we need to find a way to make predictions from this data. We can use a technique called **regression** to model the relationship between temperature and ice cream sales with an equation:

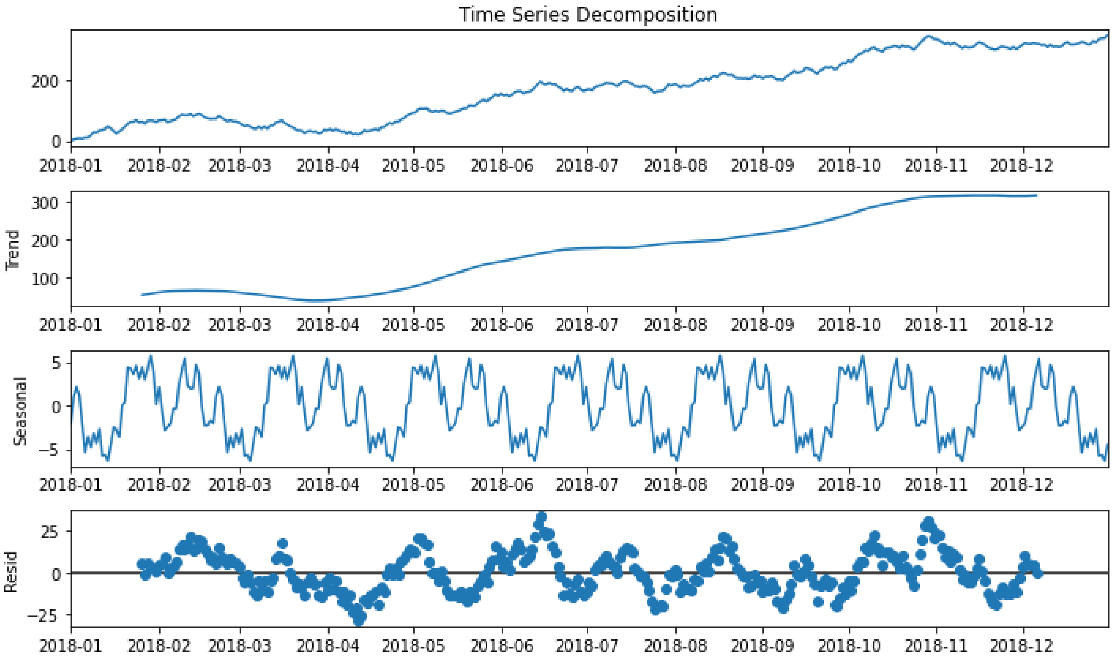
[16]:
```
ax = stats_viz.example_regression()
```



We can use the resulting equation to make predictions for the number of ice creams sold at various temperatures. However, we must keep in mind if we are interpolating or extrapolating. If the temperature value we are using for prediction is within the range of the original data we used to build our regression model, then we are **interpolating** (solid portion of the red line). On the other hand, if the temperature is beyond the values in the original data, we are **extrapolating**, which is very dangerous, since we can't assume the pattern continues indefinitely in each direction (dotted portion of the line). Extremely hot temperatures may cause people to stay inside, meaning no ice creams will be sold, while the equation indicates record-high sales.

Forecasting is a type of prediction for time series. In a process called **time series decomposition**, time series is decomposed into a trend component, a seasonality component, and a cyclical component. These components can be combined in an additive or multiplicative fashion:

```
[17]: ax = stats_viz.time_series_decomposition_example()
```



The **trend** component describes the behavior of the time series in the long-term without accounting for the seasonal or cyclical effects. Using the trend, we can make broad statements about the time series in the long-run, such as: *the population of Earth is increasing* or *the value of a stock is stagnating*. **Seasonality** of a time series explains the systematic and calendar-related movements of a time series. For example, the number of ice cream trucks on the streets of New York City is high in the summer and drops to nothing in the winter; this pattern repeats every year regardless of whether the actual amount each summer is the same. Lastly, the **cyclical** component accounts for anything else unexplained or irregular with the time series; this could be something like a hurricane driving the number of ice cream trucks down in the short-term because it isn't safe to be outside. This component is difficult to anticipate with a forecast due to its unexpected nature.

When making models to forecast time series, some common methods include ARIMA-family methods and exponential smoothing. **ARIMA** stands for autoregressive (AR), integrated (I), moving average (MA). Autoregressive models take advantage of the fact that an observation at time t is correlated to a previous observation, for example at time $t-1$. Note that not all time series are autoregressive. The integrated component concerns the differenced data, or the change in the data from one time to another. Lastly, the moving average component uses a sliding window to average the last x observations where x is the length of the sliding window. We will build an ARIMA model in chapter 7.

The moving average puts equal weight on each time period in the past involved in the calculation. In practice, this isn't always a realistic expectation of our data. Sometimes all past values are important, but they vary in their influence on future data points. For these cases, we can use exponential smoothing, which allows us to put more weight on more recent values and less weight on values further away from what we are predicting.

1.4.5 Inferential Statistics

Inferential statistics deals with inferring or deducing things from the sample data we have in order to make statements about the population as a whole. Before doing so, we need to know whether we conducted an observational study or an experiment. An observational study can't be used to determine causation because we can't control for everything. An experiment on the other hand is controlled.

Remember that the sample statistics we discussed earlier are estimators for the population parameters. Our estimators need **confidence intervals**, which provide a point estimate and a margin of error around it. This is the range that the true population parameter will be in at a certain **confidence level**. At the 95% confidence level, 95% of the confidence intervals calculated from random samples of the population contain the true population parameter.

We also have the option of using **hypothesis testing**. First, we define a null hypothesis (say the true population mean is 0), then we determine a **significance level** (1 - confidence level), which is the probability of rejecting the null hypothesis when it is true. Our result is statistically significant if the value for the null hypothesis is outside the confidence interval. More info.

```
<div style="float: left;">
    <a href="./checking_your_setup.ipynb">
        <button>Check your setup</button>
    </a>
    <a href="./python_101.ipynb">
        <button>Python 101</button>
    </a>
</div>
<div style="float: right;">
    <a href="./exercises.ipynb">
        <button>Exercises</button>
    </a>
    <a href="../ch_02/1-pandas_data_structures.ipynb">
        <button>Chapter 2 &#8594;</button>
    </a>
</div>
```